File

INHERITS FROM	Object
DECLARED IN	File.h

CLASS DESCRIPTION

The File class can be used to open a file, read information from it, write information to it, and move around within it. These facilities are, however, generic and not very rich in features. Therefore, most commonly, specialized classes will be subclassed from it.

After allocating a File object, one can ask it to open a particular file in the file system (this initalizes the object). Since a File object can work with a single file, all messages directed at it will affect the file it has opened. If asked to open a second file while it has one opened, the File object will return an error. Closing the File will free the object.

An open file in a File object has a current position, which is where any reading or writing can occurr. Reading or writing can occur in byte sized, or arbitrarily sized chunks. One can move the current position forward or backward by a specified number of bytes, or to an absolute position in the file. Writing at the end of the file will increase the file's size. All these methods that manipultate the file's contents return ErrorInfo objects when errors occurr.

One can also obtain certain information about the File, including the full pathname, just the file name, or just the basename of the file it has opened, and its total size.

This is 'beta', in so far as all the things described here are not implemented, and nothing is set in stone.
NOTE that methods return Reply objects, not plain-jane id's as suggested here!

INSTANCE VARIABLES

Inherited from Object	Class	isa;
Declared in File	Cstring	fileName;
	long int	accessType;
	FILE*	theFile;
	Integer	state;

fileName	The full pathname of the file being opened
accessType	Indicates what kind of file access we are doing
theFile	The file that the object makes use of.
state	A flag indicating what 'state' the file is in.

METHOD TYPES

Creating/Initializing, and Freeing	- openFile:For:
	- openFile:
	- closeFile
Accessing info about the file name	- getPathname

	- getPath - getFilename - getBasename - getExtension
Accessing info about the file state	- getCurrentPosition
Changing position in the file	- moveTo: - advanceBytes: - backupBytes:
Performing I/O on the file	- readByte - writeByte: - read:bytesInto:retrieving: - write:bytesFrom:writing:

CLASS METHODS

None

INSTANCE METHODS

advanceBytes:

- (id) **advanceBytes:** (FilePosDelta) *byteLoc*

Moves the current position in the file forward *byteLoc* number of bytes. If an attempt is made to move beyond the end of the file, the current position is not moved, and an error is returned.

backupBytes:

- (id) **backupBytes:** (FilePosDelta) *byteLoc*

Moves the current position backwards *byteLoc* number of bytes. If one attempts to back up beyond the beginning of the file, and error is generated, and the current position is not moved.

closeFile

- **closeFile**

Closes the files, disposes of the instance values of the method and frees the object.

fileSize

- (id) **fileSize**

Returns the size of the file in a Reply object. Gads. this means one has to know that the underlying data type is a unsigned long.....

getBasename

- (Cstring) **getBasename**

Returns the name of the file without a path or an extension. Thus, if the file name is `/usr/djb/super.eps` this will return `super`. The string that is generated must be disposed by the caller.

getCurrentPosition

- (FilePos) **getCurrentPosition**

This returns the position in the file as a byte count.

getDirectory

- (Cstring) **getDirectory**

Returns the path to the file, without including the file name itself. Note that it does include the final / in the pathname (e.g. `/usr/djb/super.eps` returns `/usr/djb/`). The string that is generated must be disposed by the caller.

getExtension

- (Cstring) **getExtension**

Like getFilename, but this returns only the extension of the file name Thus, the file `/usr/djb/super.eps` returns merely `eps`.The string that is generated must be disposed by the caller.

getFilename

- (Cstring) **getFilename**

Returns a new string that contains the name of the file without its full path. E.g. if the full name is `/usr/djb/super.eps`, this returns just `super.eps`.The string that is generated must be disposed by the caller.

getPathname

- (Cstring) **getPathname**

Returns the full pathname to the file the object has open. This provides a new string which the recipient is responsible for disposing.

initFile:For:

- (id) **initFile:** (const char *) *filename* **For:** (long int) *operation*

This acts identically to openFile:For:, below, and is provided merely to remain compatabile with what NeXT says one should provide (init menthods that start with the characters `init').

moveTo:

- (id) **moveTo:** (FilePos) *byteLoc*

Moves to the specified location in the file. If the *byteLoc* is out of range, an error is returned. If the file access method is FILE_APPEND, then this method will always return an error. If one tries to move beyond the end of a file, and error is generated (note that because FilePos is an unsigned quantity, one can not move past the beginning).

openFile:For:

- (id) **openFile:** (const char *) *filename* **For:** (long int) *operation*

Creates a new File object, opening the file *filename*, if possible, for the specified operation. The *operation* can be one of FILE_READ, FILE_WRITE, FILE_READWRITE and FILE_APPEND. In all cases save FileAPPEND, the file is opened with the current position at the beginning of the file. In the case of FileAPPEND, it is at the end of the file.

openFile:

- (id) **openFile:** (const char *) *filename*

Calls **openFile:For:** using FileREADWROTE for the *operation* parameter. This, then, simply serves as a more common case wrapper around the preceeding method.

readByte:

- (id) **readByte:** (byte*) *theByte*

Reads a byte from the file, and returns it in theByte. If an error occurrs, including detection of EOF and reads during FILE_WRITE or FILE_APPEND, an error is returned.

read:bytesInto:retrieving:

- (id) **read:** (long int) *numBytes* **bytesInto:** (byte*) *buffer* **retrieving:** (long int*) *bytesFound*

This reads up to *numBytes* into *buffer* from the file. If *buffer* is not of the proper size, the results are unpredictable. If any errors are encountered while reading, no further reading is done, and an error is returned. In all cases, the number of bytes actually read is returned in *bytesFound.*

writeByte:

- (id) **writeByte:** (byte) *theByte*

writes a byte to the file at the current position, and advances the position one byte. If an error occurs, including writing during FILE_READ, an error is returned.

write:bytesFrom:writing:

- (id) **write:** (long int) *numBytes* **bytesFrom:** (byte*) *buffer* **writing:** (long int*) *bytesWrote*

This writes up to *numBytes* from *buffer* into the file. If *buffer* is not of the proper size, the results are unpredictable. If any errors are encountered while writing, no further writing is done, and an error is returned. In all cases, the number of bytes actually written is returned in *bytesWrote.*

BUGS AND PROBLEMS

I suspect a lot of this will break when things other than plain old files are added.
there should probably be a free method corresponding to the close.
There should be a FILE_MODIFY and a FILE_READMODIFY. These are the same as write, except that write creates or clears a file to start with. Perhaps pick better terms overall. Maybe FILE_NEWWRITE, FILE_NEWREADWRITE, FILE_WRITE, FILE_READWRITE. ? That would not be backwared compatable. But this isn't 'final' anyway. Tough luck on the macpaint and macpict converters! =) Also, the error diagnostics from the open routine should reflect error codes more accurately (tried to open, but it wasn't there!)

ENHANCEMENT IDEAS

Add support for other types of streams

CONSTANTS AND DEFINED TYPES

```
/* Types of ways to access a file */
#define FILE_READ           0
#define FILE_WRITE          1
#define FILE_READWRITE      2
#define FILE_APPEND         3
```

FilePos	An unsigned number for specifying a byte position in a file.
FilePosDelta	An unsigned number for specifying a change in position in a file.

MODIFICATION HISTORY

$Log: File.rtf,v $Revision 1.5   93/04/04   23:44:30   deathSun Apr   4 23:44:30 PDT 1993Revision 1.4   93/01/10   15:07:57   deathSun Jan 10 15:07:56 PST 1993Revision 1.3 92/07/26   13:58:16   deathprobably no changes here (sigh).Revision 1.2   92/04/05   22:52:03 deathReflects   some of the miscelaneous revisions that have taken place.   Last version of version 1.Revision 1.1   92/03/29   12:19:01   deathInitial revision